**Mobile Price Range Prediction**

**Lovejeet Singh**

**Data science trainee,**

**AlmaBetter, Bangalore**

**Abstract:**

Mobile Price Range effected by many factors such as number of features a mobile device has. Or as the number of features increases the mobile price also increases. Most commonly features which may affect our mobile price range are mobile device has Bluetooth or not, mobile device has 4G or not and so on.

Our experiment and Analysis can help us to understand what could be the feature that can affects our Mobile Price Range by feature selection, feature engineering, data analysis and prediction with machine learning algorithms considering previous trends to determine the correct model.

1. **Problem Statement**

In the competitive mobile phone market companies want to understand sales data of mobile phones and factors which drive the prices. The objective is to find out some relation between features of a mobile phone (e.g.: - RAM, Internal Memory, etc.) and its selling price. In this problem, we do not have to predict the actual price but a price range indicating how high the price is.

The main objective is to build a predictive model, which could help them in predicting the price range. This would in turn help them to estimating the mobile price range or actual price.

**The company provides us dataset. The dataset holds distinctive features.**

* Data\_mobile\_price\_range.csv – holds information about mobile different features and its price range

**Following are the features insides in our dataset:**

* **Battery\_power** - Total energy a battery can store in one time measured in mAh
* **Blue** - Has bluetooth or not
* **Clock\_speed** - speed at which microprocessor executes instructions
* **Dual\_sim** - Has dual sim support or not
* **Fc** - Front Camera mega pixels
* **Four\_g** - Has 4G or not
* **Int\_memory** - Internal Memory in Gigabytes
* **M\_dep** - Mobile Depth in cm
* **Mobile\_wt** - Weight of mobile phone
* **N\_cores** - Number of cores of processor
* **Pc** - Primary Camera mega pixels
* **Px\_height** - Pixel Resolution Height
* **Px\_width** - Pixel Resolution Width
* **Ram** - Random Access Memory in Mega Bytes
* **Sc\_h** - Screen Height of mobile in cm
* **Sc\_w** - Screen Width of mobile in cm
* **Talk\_time** - longest time that a single battery charge will last when you are
* **Three\_g** - Has 3G or not
* **Touch\_screen** - Has touch screen or not
* **Wifi** - Has wifi or not
* **Price\_range** - This is the target variable with value of
  + 0(low cost),
  + 1(medium cost),
  + 2(high cost) and
  + 3(very high cost).

**Following are the library we will use in our analysis and model building: -**

* Pandas: - Pandas is for solving data wrangling and exploratory.
* NumPy: - NumPy is for numerical problem solving
* Matplotlib: - Matplotlib is for Data Visualization
* Seaborn: - Seaborn is for Data Visualization
* Scikit Learn: - Scikit learn for Machine learning

1. **Steps involved:**

* **Exploratory Data Analysis**

After loading the datasets, we performed this method by comparing our target variable that is Price Range with other independent variables. This process helped us figuring out various aspects, distribution and relationships among the target and the independent variables. It gave us a better idea of which feature behaves in which manner compared to the target variable.

* **Null values Treatment**

In Our dataset there are two features px\_height and sc\_w has null/zero values, in which px\_height has 180 observations has value as zero while in sc\_w only has 2 observation which contains zero value. For sc\_w feature we decided to simple delete those observation which has value zero. For px\_height feature One way is, we replace them using the mean, mode and median of these features according to their distribution. And other way is by using KNNImputer we replace those observation which has zero value to their nearest neighbor.

* **Feature Selection**

The assumption of Logistic Regression classifier says that the multicollinearity of independent feature must be low that is why we use Variance Inflation Factor (VIF) to check the multicollinearity of each feature.

For the other algorithm like Decision tree, random forest etc. these algorithms automatically manage multicollinearity and correlation. So, for these algorithms we will use all the dataset no matter their VIF in high or low.

* **Standardization of features**

Our main motive through this step was to scale our data into a uniform format that would allow us to utilize the data in a better way while performing fitting and applying different algorithms to it. To scale our data, we use StandardScaler which is available in preprocessing class present in scikit learn library. This step is more important when we use algorithm like Logistic Regression Classifier and Support Vector Machine.

* **Fitting different models**

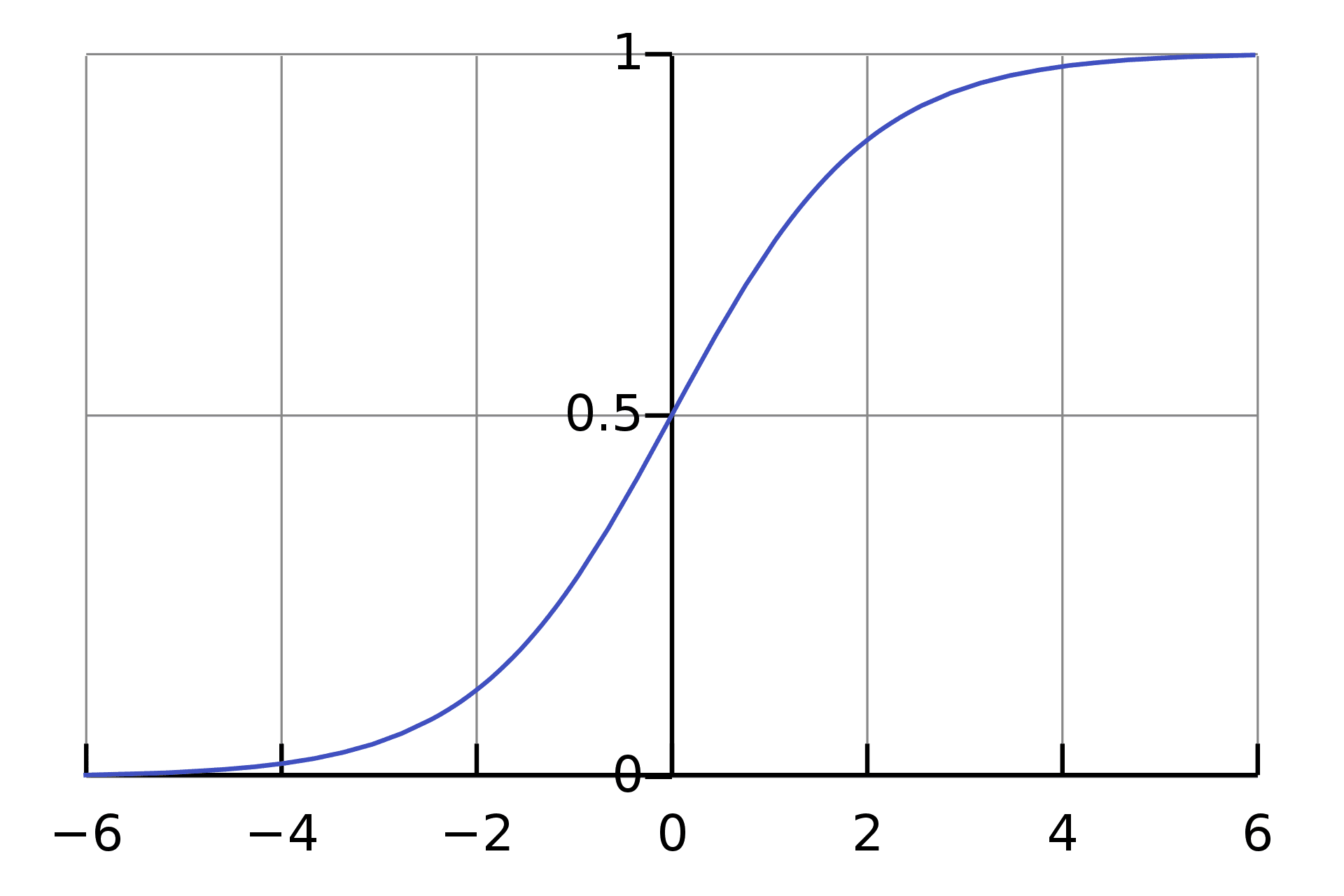
For modelling we tried various Classification algorithms like:

* 1. Logistic Regression Classifier
  2. Decision Tree
  3. Random Forest
  4. XGBoost
  5. Gradient Boosting
  6. Support Vector Machine
* **Tuning the hyperparameters for better accuracy**

Tuning the hyperparameters of respective algorithms is necessary for getting better performance and to avoid overfitting in case of tree-based models like Random Forest Regressor, XGBoost Regressor and Gradient Boosting.

1. **Algorithms:**
2. **Logistic Regression Classifier**

Logistic Regression is a statistical approach and a Machine Learning algorithm that is used for classification problems and is based on the concept of probability. It is used when the dependent variable (target) is categorical.

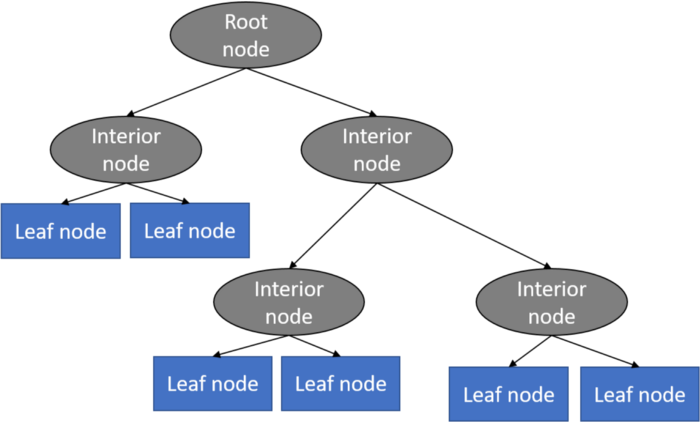


This is an easy way to identify the Sigmoid function or the logistic function.  
In regard to Logistic Regression, the concept used is the threshold value. The threshold values help to define the probability of either 0 or 1. For example, values above the threshold value tend to 1, and a value below the threshold value tends to 0.

**Type of Logistic Regression Classifier**

1. **Binomial:** This means that there can be only two possible types of the dependent variables, such as 0 or 1, Yes or No, etc.
2. **Multinomial:** This means that there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
3. **Ordinal:** This means that there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".
4. **Decision Tree Classifier**

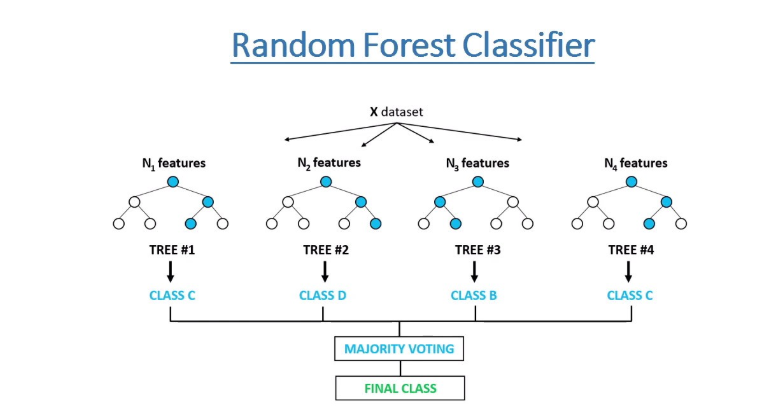
Decision Tree is one of the most used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application.



It is a tree-structured classifier with three types of nodes. The Root Node is the initial node which represents the entire sample and may get split further into further nodes. The Interior Nodes represent the features of a data set and the branches represent the decision rules. Finally, the Leaf Nodes represent the outcome. This algorithm is very useful for solving decision-related problems.

1. **Random Forest Classifier**

The core unit of random forest classifiers is the decision tree. The decision tree is a hierarchical structure that is built using the features (or the independent variables) of a data set. Each node of the decision tree is split according to a measure associated with a subset of the features. The random forest is a collection of decision trees that are associated with a set of bootstrap samples that are generated from the original data set. The nodes are split based on the entropy (or Gini index) of a selected subset of the features. The subsets that are created from the original data set, using bootstrapping, are of the same size as the original data set.



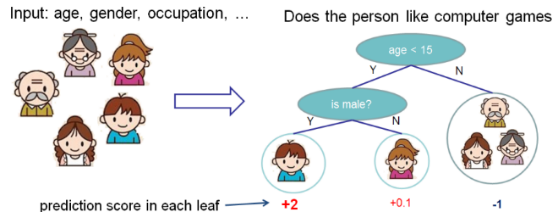
In the standard random forest approach, the bootstrapping technique helps the development of random forest with a set of required number of decision trees to improve classification accuracy through the concept of overlap thinning as mentioned in Suthaharan (2015). Then an approach called bagging (bootstrap aggregate) technique is used to select the best trees with a voting scheme. This standard random forest approach is the one adopted in the proposed cognitive computing architecture.

1. **XGBoost Classifier**

XGBoost is an implementation of Gradient Boosted decision trees. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

1. **Gradient Boosting Classifier**

Gradient boosted trees consider the special case where the simple model is a decision tree.

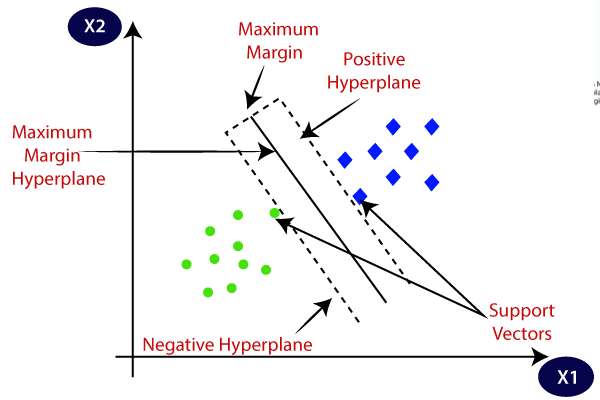


In this case, there are going to be 2 kinds of parameters P: the weights at each leaf, w, and the number of leaves T in each tree (so that in the above example, T=3 and w= [2, 0.1, -1]).

When building a decision tree, a challenge is to decide how to split a current leaf. For instance, in the above image, how could I add another layer to the (age > 15) leaf? A ‘greedy’ way to do this is to consider every possible split on the remaining features (so, gender and occupation), and calculate the new loss for each split; you could then pick the tree which most reduces your loss.

1. **Support Vector Machine**

SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane**.**

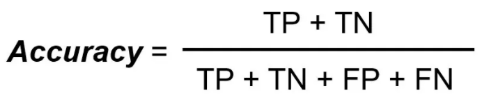
****

1. **Model performance:**

Model can be evaluated by various metrics such as:

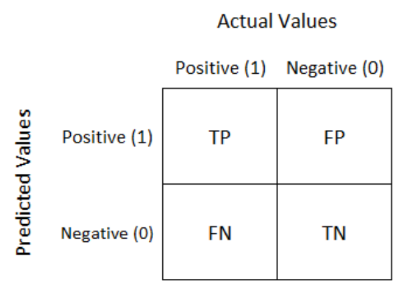
* 1. Accuracy

Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.



* 1. Confusion Matrix

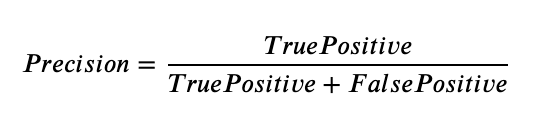
Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values



The other metrics of the confusion matrix

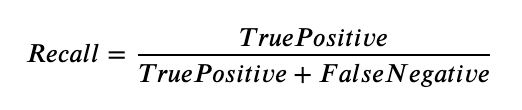
* + 1. Precision:

Precision for a label is defined as the number of true positives divided by the number of predicted positives.



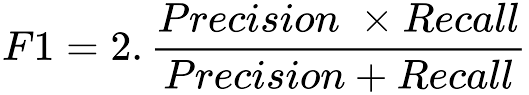
* + 1. Recall (Sensitivity):

Recall for a label is defined as the number of true positives divided by the total number of actual positives.



* + 1. F1 Score:

It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.



* + 1. AUC-ROC:

The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR (True Positive Rate) against the FPR (False Positive Rate) at various threshold values and separates the ‘signal’ from the ‘noise’.

1. **Hyper parameter tuning**

Hyperparameters are sets of information that are used to control the way of learning an algorithm. Their definitions impact parameters of the models, seen as a way of learning, change from the new hyperparameters. This set of values affects performance, stability, and interpretation of a model. Each algorithm requires a specific hyperparameters grid that can be adjusted according to the business problem. Hyperparameters alter the way a model learns to trigger this training algorithm after parameters to generate outputs. We used Grid Search CV

* 1. **Grid Search CV-**Grid Search combines a selection of hyperparameters established by the scientist and runs through all of them to evaluate the model’s performance. Its advantage is that it is a simple technique that will go through all the programmed combinations. The biggest disadvantage is that it traverses a specific region of the parameter space and cannot understand which movement or which region of the space is important to optimize the model

1. **Conclusion:**

That is, it! We reached the end of our exercise. Starting with loading the data so far, we have done EDA, null values treatment, encoding of categorical columns, feature selection and then model building. In all these models our accuracy revolves in the range of 69%

to 100% on train set and 51% to 95% on test set.

So, the performance of our best model Support Vector Machine which give as accuracy score is 96% which can be said to be great for this dataset. So, we can deploy this model for solve business problem.

Images Source: Google , Towardsdatascience, Greeksforgeeks